The VLSI Design of a Reed-Solomon Encoder Using Berlekamp's Bit-Serial Multiplier Algorithm

T. K. Truong and L. J. Deutsch
Communications Systems Research Section

I. S. Reed, I.-S. Hsu, K. Wang, and C.-S. Yeh
University of Southern California

E. R. Berlekamp has developed for the Jet Propulsion Laboratory a bit-serial multiplication algorithm for the encoding of Reed-Solomon (RS) codes, using a dual basis over a Galois field. The conventional RS-encoder for long codes often requires look-up tables to perform the multiplication of two field elements. Berlekamp's algorithm requires only shifting and exclusive-OR operations. It is shown in this paper that the new dual-basis (255, 223) RS-encoder can be realized readily on a single VLSI chip with NMOS technology.

I. Introduction

A concatenated Reed-Solomon/Viterbi channel encoding system has been suggested both by the European Space Agency (ESA) (Ref. 1) and JPL (Refs. 2, 3) for the deep-space downlink. The standard RS-encoder design developed by JPL assumes the following codes and parameters.

Let $GF(2^m)$ be a finite field. Then an RS code is a sequence of the symbols in $GF(2^m)$. This sequence of symbols can be considered to be the coefficients of a polynomial. The code polynomial of such a code is

$$C(x) = \sum_{i=0}^{n-1} c_i x^i \tag{1}$$

wnere $c_i \in GF(2^m)$.

The parameters of an RS code are summarized as follows:

m = number of bits per symbol

$n = 2^m - 1$ = the length of a codeword in symbols

t = maximum number of error symbols that can be corrected

$d = 2t + 1$ = design distance

$2t$ = number of check symbols

$k = n - 2t$ = number of information symbols

In the JPL design, $m = 8$, $n = 255$, $t = 16$, $d = 33$, $2t = 32$, and $k = 223$. This code is the (255, 223) RS code.

The generator polynomial of an RS code is defined by

$$g(x) = \sum_{j=b}^{b+2t-1} (x - \gamma^j) = \sum_{i=0}^{2t} g_i x^i \qquad (2)$$

where b is a nonnegative integer, usually chosen to be 1, and γ is a primitive element in $GF(2^m)$. In order to reduce the complexity of the encoder it is desirable to make the coefficients of $g(x)$ symmetric so that $g(x) = x^{-d-1}\, g(1/x)$. To accomplish this b must be chosen to satisfy $2b + d - 2 = 2^m - 1$. Thus for the JPL code $b = 112$.

Let $I(x) = c_{2t}x^{2t} + c_{2t+1}x^{2t+1} + \cdots + c_{n-1}x^{n-1}$ and $P(x) = c_0 + c_1 x + \cdots + c_{2t-1}x^{2t-1}$ be the information polynomial and the check polynomial, respectively. Then the encoded RS code polynomial is represented by

$$C(x) = I(x) + P(x) \qquad (3)$$

To be an RS code $C(x)$ must be also a multiple of $g(x)$. That is,

$$C(x) = q(x)g(x) \qquad (4)$$

To find $P(x)$ in Eq. (3) such that Eq. (4) is true, divide $I(x)$ by $g(x)$. The division algorithm yields

$$I(x) = q(x)g(x) + r(x) \qquad (5)$$

Also let $r(x) = -P(x)$, then by Eq. (5)

$$q(x)g(x) = I(x) - r(x) = I(x) + P(x) = C(x) \qquad (6)$$

Figure 1 shows the structure of a t-error correcting RS encoder over $GF(2^m)$. In Fig. 1 R_i for $0 \leqslant i \leqslant 2t - 1$ and Q are m-bit registers. Initally all registers are set to zero, and both switches (controlled by a control signal SL) are set to position A.

The information symbols $c_{n-1}, \cdots, c_{2t}$ are fed into the division circuit of the encoder and also transmitted out of the encoder one by one. The quotient coefficients are generated and loaded into Q register sequentially. The remainder coefficients are computed successively. Immediately after c_{2t} is fed to the circuit, both switches are set to position B. At the very same moment c_{2t-1} is computed and transmitted. Simultaneously, c_i is being computed and loaded into register R_i for $0 \leqslant i \leqslant 2t - 2$. Next $c_{2t-2}, \cdots, c_0$ are transmitted out of the encoder one by one. $c_{2t-2}, \cdots, c_0$ retain their values because the content of Q is set to zero when the upper switch is at position B.

The complexity of the design of an RS encoder results from the computation of products zg_i for $0 \leqslant i \leqslant 2t - 2$. These computations can be performed in several ways (Ref. 3). Unfortunately none of them is suited to the pipeline processing structures usually seen in VLSI design. Recently, Berlekamp (Ref. 4) developed a bit-serial multiplier algorithm that has the features needed to solve this problem. Perlman and Lee (Ref. 5) show in detail the mathematical basis for this algorithm. In this paper Berlekamp's method is applied to the VLSI design of a (255, 223) RS-encoder, which can be implemented on a single VLSI chip.

II. Berlekamp's Bit-Serial Multiplier Algorithm Over *GF*(2^m)

In order to understand Berlekamp's multiplier algorithm some mathematical preliminaries are needed. Toward this end the mathematical concepts of the trace and a complementary (or dual) basis are introduced. For more details and proofs see Refs. 3, 4 and 5.

Definition 1: The trace of an element β belonging to $GF(p^m)$, the Galois field of P^m elements, is defined as follows:

$$Tr(\beta) = \sum_{k=0}^{m-1} \beta^{p^k}$$

In particular, for $p = 2$,

$$Tr(\beta) = \sum_{k=0}^{m-1} (\beta)^{2^k}$$

The trace has the following properties

(1) $[Tr(\beta)]^p = \beta + \beta^p + \cdots + \beta^{p^{m-1}} = Tr(\beta)$, where $\beta \in GF(p^m)$. This implies that $Tr(\beta) \in GF(p)$; i.e., the trace is on the ground field $GF(p)$.

(2) $Tr(\beta + r) = Tr(\beta) + Tr(r)$, where $\beta, r \in GF(p^m)$

(3) $Tr(c\beta) = cTr(\beta)$, where $c \in GF(p)$.

(4) $Tr(1) \equiv m (\mathrm{mod}\ p)$.

Definition 2: A basis $\{\mu_j\}$ in $GF(p^m)$ is a set of m linearly independent elements in $GF(p^m)$.

Definition 3: Two bases $\{\mu_j\}$ and $\{\lambda_k\}$ are said to be complementary or the dual of one another if

$$Tr(\mu_j \lambda_k) = \begin{cases} 1, & \text{if } j = k \\ 0, & j \neq k \end{cases}$$

The basis $\{\mu_j\}$ is called the original basis, and the basis $\{\lambda_k\}$ is called the dual basis.

Lemma: If α is a root of an irreducible polynomial of degree m in $GF(p^m)$, then $\{\alpha^k\}$ for $0 \leqslant k \leqslant m-1$ is a basis of $GF(p^m)$. The basis $\{\alpha^k\}$ for $0 \leqslant k \leqslant m-1$ is called the normal or natural basis of $GF(p^m)$.

Theorem 1 (Theorem 19 in Ref. 4): Every basis has a complementary basis.

Corollary 1: Suppose the bases $\{\mu_j\}$ and $\{\lambda_k\}$ are complementary. Then a field element z can be expressed in the dual basis $\{\lambda_k\}$ by the expansion

$$z = \sum_{k=0}^{m-1} z_k \lambda_k = \sum_{k=0}^{m-1} Tr(z\mu_k)\lambda_k$$

where $z_k = Tr(z\mu_k)$ is the kth coefficient of the dual basis.

Proof: Let $z = z_0\lambda_0 + z_1\lambda_1 + \cdots + z_{m-1}\lambda_{m-1}$. Multiply both sides by α^k and take the trace. Then by Def. 3 and the properties of the trace,

$$Tr(z\alpha^k) = Tr\left(\sum_{i=0}^{m-1} z_i(\lambda_i\mu_k)\right) = z_k \qquad \text{Q.E.D.}$$

The following corollary is an immediate consequence of Corollary 1.

Corollary 2: The product $w = zy$ of two field elements in $GF(p^m)$ can be expressed in the dual basis by the expansion

$$w = \sum_{k=0}^{m-1} Tr(zy\mu_k)\lambda_k$$

where $Tr(zy\mu_k)$ is the kth coefficient of the dual basis for the product of two field elements.

These two corollaries provide a theoretical basis for the new RS-encoder algorithm.

III. A Simple Example of Berlekamp's Algorithm Applied to an RS-Encoder

This section follows the treatment in Ref. 3. It is included here for two purposes. First, Ref. 3 is not readily available for most readers. Second, this example is included to illustrate how Berlekamp's new bit-serial multiplier algorithm can be used to realize the RS-encoder structure presented in Fig. 1.

Consider a (15, 11) RS code over $GF(2^4)$. For this code, $m = 4$, $n = 15$, $t = 2$, $d = 2t + 1 = 5$, and $n - 2t = 11$ information symbols. Let α be a root of the primitive irreducible polynomial $f(x) = x^4 + x + 1$ over $GF(2)$. α satisfies $\alpha^{15} = 1$. An element z in $GF(2^4)$ is representable by 0 or α^j for some j, $0 \leqslant j \leqslant 14$. z can be represented also by a polynomial in α over $GF(2)$. This is the representation of $GF(2^4)$ in the normal basis $\{\alpha^k\}$ for $0 \leqslant k \leqslant 3$. That is, $z = u_0 + u_1\alpha + u_2\alpha^2 + u_3\alpha^3$, where $u_k \in GF(2)$ for $0 \leqslant k \leqslant 3$.

In Table 1, the first column is the index or logarithm of an element in base α. The logarithm of the zero element is denoted by an asterisk. Columns 2 to 5 show the 4-tuples of the coefficients of the elements expressed as polynomials.

The trace of an element z in $GF(2^4)$ is found by Def. 1 and the properties of the trace to be

$$TR(z) = u_0 Tr(1) + u_1 Tr(\alpha) + u_2 Tr(\alpha^2) + u_3 Tr(\alpha^3)$$

where $Tr(1) \equiv 4 \pmod 2 = 0$, $Tr(\alpha) = Tr(\alpha^2) = \alpha + \alpha^2 + \alpha^4 + \alpha^8 = 0$ and $Tr(\alpha^3) = \alpha^3 + \alpha^6 + \alpha^9 + \alpha^{12} = 1$. Thus $Tr(z) = u_3$. The trace element α^k in $GF(2^4)$ is listed in column 3 of Table 1.

By Def. 2 any set of four linearly independent elements can be used as a basis for the field $GF(2^4)$. To find the dual basis of the normal basis $\{\alpha^j\}$ in $GF(2^4)$ let a field element z be expressed in dual basis $\{\lambda_0, \lambda_1, \lambda_2, \lambda_3\}$. From Corollary 1 the coefficients of z are $z_k = Tr(z\alpha^k)$ for $0 \leqslant k \leqslant 3$. Thus $z_0 = Tr(z)$, $z_1 = Tr(z\alpha)$, $z_2 = Tr(z\alpha^2)$ and $z_3 = Tr(z\alpha^3)$. Let $z = \alpha^i$ for some i, $0 \leqslant i \leqslant 14$. Thus a coefficient z_k, for $0 \leqslant k \leqslant 3$, of an element z in the dual space can be obtained by cyclically shifting the trace column in Table 1 upward by k positions where the first row is excluded. These appropriately shifted columns of coefficients are shown in Table 1 as the last four columns. In Table 1 the elements of the dual basis, $\lambda_0, \lambda_1, \lambda_2, \lambda_3$, are underlined. Evidently $\lambda_0 = \alpha^{14}$, $\lambda_1 = \alpha^2$, $\lambda_2 = \alpha$ and $\lambda_3 = 1$ are the four elements of the dual basis.

In order to make the generator polynomial $g(x)$ symmetric b must satisfy the equation $2b + d - 2 = 2^m - 1$. Thus $b = 6$ for this code. The γ in Eq. (2) can be any primitive element in $GF(2^4)$. It will be shown in Section IV that γ can be chosen to simplify the binary mapping matrix. In this example let $\gamma = \alpha$. Thus the generator polynomial is given by

$$g(x) = \prod_{j=6}^{9} (x - \alpha^j) = \sum_{i=0}^{4} g_1 x^i \tag{7}$$

where $g_0 = g_4 = 1, g_1 = g_3 = \alpha^3$ and $g_2 = \alpha$.

Let g_i be expressed in the normal basis $\{1, \alpha, \alpha^2, \alpha^3\}$. Let z, a field element, be expressed in the dual basis; i.e., $z = z_0\lambda_0 + z_1\lambda_1 + z_2\lambda_2 + z_3\lambda_3$. In Fig. 1 the products zg_i for $0 \leqslant i \leqslant 3$ needs to be computed.

Since $g_3 = g_1$, it is necessary to compute only zg_0, zg_1 and zg_2. Let the products zg_i for $0 \leqslant i \leqslant 2$ be represented in the dual basis. By Corrollary 2 zg_i can be expressed in the dual basis as

$$z \begin{bmatrix} g_0 \\ g_1 \\ g_2 \end{bmatrix} = \sum_{k=0}^{3} \begin{bmatrix} T_0^{(k)}(z) \\ T_1^{(k)}(z) \\ T_2^{(k)}(z) \end{bmatrix} \lambda_k \tag{8}$$

where $T_i^{(k)}(z) = Tr(zg_i\alpha^k)$ is the kth coefficient (or kth bit) of zg_i for $0 \leqslant i \leqslant 2$ and $0 \leqslant k \leqslant 3$.

The present problem is to express $T_i^{(k)}(z)$ recursively in terms of $T_i^{(k-1)}(z)$ for $1 \leqslant k \leqslant 3$. Initially for $k = 0$,

$$\begin{bmatrix} T_0^{(0)}(z) \\ T_1^{(0)}(z) \\ T_2^{(0)}(z) \end{bmatrix} = \begin{bmatrix} Tr(zg_0) \\ Tr(zg_1) \\ Tr(zg_2) \end{bmatrix} = \begin{bmatrix} Tr(z\alpha^0) \\ Tr(z\alpha^3) \\ Tr(z\alpha) \end{bmatrix} = \begin{bmatrix} z_0 \\ z_3 \\ z_1 \end{bmatrix} \tag{9}$$

where $TR(z\alpha^j) = Tr((z_0\lambda_0 + z_1\lambda_1 + z_2\lambda_2 + z_3\lambda_3)\alpha^j) = z_j$ for $0 \leqslant j \leqslant 3$. Equation (9) can be expressed in a matrix form as follows:

$$\begin{bmatrix} T_0^{(0)}(z) \\ T_1^{(0)}(z) \\ T_2^{(0)}(z) \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{10}$$

The above matrix is the 3×4 binary mapping matrix of the problem.

To compute $T_i^{(k)}(z)$ *for* $k > 0$, observe that $T_i^{(k)}(z) = Tr((\alpha z)g_i\alpha^{k-1}) = T^{(k-1)}(\alpha z)$. Hence $T_i^{(k)}(z)$ is obtained from $T_i^{(k-1)}(z)$ by replacing z by $y = \alpha z$. Let $\alpha z = y = y_0\lambda_0 + y_1\lambda_1 + y_2\lambda_2 + y_3\lambda_3$, where $y_m = Tr(y\alpha^m) = Tr(z\alpha^{m+1})$ for $0 \leqslant m \leqslant 3$. Then $T_i^{(k)}$ is obtained from $T_i^{(k-1)}$ by replacing z_0 by $y_0 = Tr(z\alpha) = z_1$, z_1 by $y_1 = Tr(z\alpha^2) = z_2$, z_2 by $y_2 = Tr(z\alpha^3) = z_3$ and z_3 by $y_3 = Tr(z\alpha^4) = Tr(z(\alpha + 1)) = z_0 + z_1$.

To recapitulate $zg_i = T_i^{(0)}\lambda_0 + T_i^{(1)}\lambda_1 + T_i^{(2)}\lambda_2 + T_i^{(3)}\lambda_3$, where $0 \leqslant i \leqslant 3$ and $z = z_0\lambda_0 + z_1\lambda_1 + z_2\lambda_2 + z_3\lambda_3$, can be computed by Berlekamp's bit-serial multiplier algorithm as follows:

(1) Initially for $k = 0$, compute $T_0^{(0)}(z)$, $T_1^{(0)}(z)$ and $T_2^{(0)}(z)$ by Eq. (10). Also $T_3^{(0)}(z) = T_1^{(0)}(z)$.

(2) For $k = 1, 2, 3$, compute $T_i^{(k)}(z)$ by

$$T_i^{(k)}(z) = T_i^{(k-1)}(y)$$

where $0 \leqslant i \leqslant 3$ and $y = \alpha z = y_0\lambda_0 + y_1\lambda_1 + y_2\lambda_2 + y_3\lambda_3$ with $y_0 = z_1, y_1 = z_2, y_2 = z_3$ and $y_3 = z_0 + z_1 = T_f$, where $T_f = z_0 + z_1$ is the feedback term of the algorithm.

The above example illustrates Berlekamp's bit-serial multiplier algorithm. This algorithm developed in Refs. 4 and 5 requires shifting and XOR operations only. Berlekamp's dual basis RS-encoder is well-suited to a pipeline structure which can be implemented in VLSI design. The same procedure extends similarly to the design of a (255, 223) RS-encoder over $GF(2^8)$.

IV. A VLSI Architecture of a (255, 223) RS-Encoder with Dual-Basis Multiplier

In this section an architecture is designed to implement (255, 223) RS-encoder using Berlekamp's multiplier algorithm. The circuit is a direct mapping from an encoder using Berlekamp's bit-serial algorithm as developed in the previous sections to an architectural design. This architecture can be realized quite readily on a single NMOS VLSI chip.

Let $GF(2^8)$ be generated by α, where α is a root of a primitive irreducible polynomial $f(x) = x^8 + x^7 + x^2 + x + 1$ over $GF(2)$. The normal basis of this field is $\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7\}$. The representations of this field in both the normal basis and its dual basis are tabulated in Appendix A. From Corrollary 1 the coefficients of a field element α^j can be obtained from $z_k = Tr(\alpha^{j+k})$ for $0 \leqslant k \leqslant 7$, where $\alpha^j = z_0\lambda_0 + \cdots + z_7\lambda_7$. From Table A-1 in Appendix A, the dual basis

$\{\lambda_0, \lambda_1, \cdots, \lambda_7\}$ of the normal basis is the ordered set $\{\alpha^{99}, \alpha^{197}, \alpha^{203}, \alpha^{202}, \alpha^{201}, \alpha^{200}, \alpha^{199}, \alpha^{100}\}$.

It was mentioned previously that γ in Eq. (2) can be chosen to simplify the binary mapping matrix. Two binary matrices, one for the primitive element $\gamma = \alpha^{11}$ and the other for $\gamma = \alpha$, were computed. It was found that the binary mapping matrix for $\gamma = \alpha^{11}$ had a smaller number of 1's. Hence this binary mapping matrix was used in the design. For this case the generator polynomial $g(x)$ of the RS-encoder over $GF(2^8)$ was given by

$$g(x) = \prod_{j=112}^{143} (x - \alpha^{11j}) = \sum_{i=0}^{32} g_i x^i \tag{11}$$

where $g_0 = g_{32} = 1, g_1 = g_{31} = \alpha^{249}, g_2 = g_{30} = \alpha^{59}, g_3 = g_{29} = \alpha^{66}, g_4 = g_{28} = \alpha^4, g_5 = g_{27} = \alpha^{43}, g_6 = g_{26} = \alpha^{126}, g_7 = g_{25} = \alpha^{251}, g_8 = g_{24} = \alpha^{97}, g_9 = g_{23} = \alpha^{30}, g_{10} = g_{22} = \alpha^3, g_{11} = g_{21} = \alpha^{213}, g_{12} = g_{20} = \alpha^{50}, g_{13} = g_{19} = \alpha^{66}, g_{14} = g_{18} = \alpha^{170}, g_{15} = g_{17} = \alpha^5$, and $g_{16} = \alpha^{24}$.

The binary mapping matrix for the coefficients of the generator polynomial in Eq. (11) is computed and shown in Appendix B. The feedback term T_f in Berlekamp's algorithm is found in this case to be:

$$T_f = Tr(\alpha^8 z) = Tr((\alpha^7 + \alpha^2 + \alpha + 1)z) = z_0 + z_1 + z_2 + z_7 \tag{12}$$

In the following a VLSI chip architecture is designed to realize a (255, 223) RS-encoder using the above parameters and Berlekamp's algorithms. An overall block diagram of this chip is shown in Fig. 2. In Fig. 2 VDD and GND are power pins. CLK is a clock signal, which in general is a periodic square wave. The information symbols are fed into the chip from the data-in pin DIN bit-by-bit. Similarly the encoded codeword is transmitted out of the chip from the data-out pin DOUT sequentially. The control signal LM (load mode) is set to 1 (logic 1) when the information symbols are loaded into the chip. Otherwise, LM is set to 0.

The input data and LM signals are synchronized by the CLK signal, while the operations of the circuit and output data signal are synchronized by two nonoverlapping clock signals $\phi 1$ and $\phi 2$. To save space, dynamic registers are used in this design. A logic diagram of a 1-bit dynamic register with reset is shown in Fig. 3. The timing diagram of CLK, $\phi 1$, $\phi 2$, LM, DIN and DOUT signals are shown in Fig. 4. The delay of DOUT with respect to DIN is due to the input and output flip-flops.

Figure 5 shows the block diagram of a (255, 223) RS-encoder over $GF(2^8)$ using Berlekamp's bit-serial multiplier algorithm. The circuit is divided into five units. The circuits in each unit are discussed in the following:

(1) Product Unit: The Product Unit is used to compute T_f, $T_{31}, \cdots, T_0$. This circuit is realized by a Programmable Logic Array (PLA) circuit [6]. Since $T_0 = T_{31}$, $T_1 = T_{30}, \cdots, T_{15} = T_{17}$, only $T_f, T_{31}, \cdots, T_{17}$ and T_{16} are actually implemented in the PLA circuit. $T_0, \cdots, T_{15}$ are connected directly to $T_{31}, \cdots, T_{17}$, respectively. Over other circuits a PLA circuit has the advantage of being easy to reconfigure.

(2) Remainder Unit: The Remainder Unit is used to store the coefficients of the remainder during the division process. In Fig. 5, S_i for $0 \leqslant i \leqslant 30$ are 8-bit shift registers with reset. The addition in the circuit is a modulo 2 addition or Exclusive-OR operation. While c_{32} is being fed to the circuit, c_{31} is being computed and transmitted sequentially from the circuit. Simultaneously c_i is computed and then loaded into S_i for $0 \leqslant i \leqslant 30$. Then $c_{30}, \cdots, c_0$ are transmitted out of the encoder bit-by-bit.

(3) Quotient Unit: In Fig. 5, Q and R represent a 7-bit shift register with reset and an 8-bit shift register with reset and parallel load, respectively. R and Q store the currently operating coefficient and the next coefficient of the quotient polynomial, respectively. A logic diagram of register R is shown in Fig. 6. z_i is loaded into R_i every eight clock cycles, where $0 \leqslant i \leqslant 7$. Immediately after all 223 information symbols are fed into the circuit, the control signal SL changes to logic 0. Thenceforth the contents of Q and R are zero so that the check symbols in the Remainder Unit sustain their values.

(4) I/O Unit: This unit handles the input/output operations. In Fig. 5 both F_0 and F_1 are flip-flops. A pass transistor controlled by $\phi 1$ is inserted before F_1 for the purpose of synchronization. Control signal SL selects whether a bit of an information symbol or a check symbol is to be transmitted.

(5) Control Unit: The Control Unit generates the necessary control signals. This unit is further divided into 3 portions, as shown in Fig. 7. The two-phase clock generator circuit in Ref. 6 is used to convert a clock signal into two nonoverlapping clock signals $\phi 1$ and $\phi 2$. In Fig. 8 is shown a logic diagram of the circuit for generating control signals START and SL. Control signal START resets all registers and the divide-by-8

counter before the encoding process begins. Control signal LD is simply generated by a divide-by-8 counter to load the z_i's into the R_i's in parallel.

Since a codeword contains 255 symbols, the computation of a complete encoded codeword requires 255 "symbol cycles." A symbol cycle is the time interval for executing a complete cycle of Berlekamp's algorithm. Since a symbol has 8 bits, a symbol cycle contains 8 "bit cycles." A bit cycle is the time interval for executing one step in Berlekamp's algorithm. In this design a bit cycle requires a period of the clock cycle.

The layout design of this (255, 223) RS-encoder is shown in Fig. 9. Before the design of the layout each circuit was simulated on a general-purpose computer by using SPICE (a transistor-level circuit simulation program) (Ref. 7). The circuit requires about 3000 transistors, while a similar JPL design requires 30 CMOS IC chips (Ref. 5). This RS-encoder design will be fabricated and tested in the near future.

V. Concluding Remarks

A VLSI structure is developed for a Reed-Solomon encoder using Berlekamp's bit-serial multiplier algorithm. This structure is both regular and simple.

The circuit in Fig. 2 can be modified easily to encode an RS code with a different field representation and different parameters other than those used in Section IV. Table 2 shows the primary modifications needed in the circuit to change a given parameter.

Acknowledgment

The authors wish to acknowledge Marvin Perlman of the Information Systems Engineering Section of JPL. His initial work with E. Berlekamp eventually resulted in the acceptance of the Berlekamp code as a guideline for both NASA and ESA. Also, his tutorial paper on the algorithm (Reference 5) was particularly helpful to the authors.

References

1. Reefs, H. F. A., and Best, A. R., "Concatenated Coding on A Spacecraft-to-Ground Telemetry Channel Performance," Proc. ICC 81, Denver, 1981.

2. Odenwader, J., et al., "Hybrid Coding System Study Final Report," Linkabit Corp., NASA CR114, 486, September 1972.

3. MacWilliams, P. J., and Sloane, N. J. A., *The Theory Of Error-Correcting Codes*, North-Holand Publishing Company, 1978.

4. Berlekamp, E. R., "Technical Proposal for A Low-Power Reed-Solomon Encoder/Interleaver Using About 30 CMOS IC's," Cyclotomics, Inc., in response to RFP No. BP-6-9007.

5. Perlman, M. and Lee, J. J., "A Comparison of Conventional Reed-Solomon Encoders and Berlekamp's Architecture," New Technology Report NPO-15568, March 10, 1982, Case No. D-15568, NASA.

6. Mead, C., and Conway, L., *Introduction To VLSI Systems*, Addison-Wesley Publishing Company, Calif., 1980.

7. Negal, L. W., and Pederson, D. O., "SPICE – Simulation Program with Integrated Circuit Emphasis," Memorandum No. ERL-M382, Electronics Research Laboratory, University of California, Berkeley, April 12, 1973.

Table 1. Representations of elements over $GF(2^4)$ generated by $\alpha^4 = \alpha + 1$

Power j	Elements in normal base	$Tr(\alpha^j)$	Elements in dual base
	$\alpha^3\alpha^2\alpha^1\alpha^0$		$z_0z_1z_2z_3$
*	0000	0	0000
0	0001	0	0001 λ_3
1	0010	0	0010 λ_2
2	0100	0	0100 λ_1
3	1000	1	1001
4	0011	0	0011
5	0110	0	0110
6	1100	1	1101
7	1011	1	1010
8	0101	0	0101
9	1010	1	1011
10	0111	0	0111
11	1110	1	1111
12	1111	1	1110
13	1101	1	1100
14	1001	1	1000 λ_0

Table 2. The primary modifications of the encoder circuit in Fig. 2 needed to change a parameter

Parameter to be changed	The value used for the circuit in Fig. 2	New value	The circuits of Fig. 2 that require modification
1. Generator polynomial	Eq. (8)	$g(x)$	The PLA of the Product Unit needs to be changed
2. The finite field used	$GF(2^8)$	$GF(2^m)$	All registers are m-bit resistors, except Q is a $(m - 1)$-bit register. A divide-by-m counter is used. (The generator polynomial may not be changed.)
3. Error-correcting capability	16	t	$2t$-2 shift registers are required in the Remainder Unit. (The generator polynomial is also changed.)
4. Number of information symbols	223	k	None is changed, since k is implicitly contained in the control signal LM

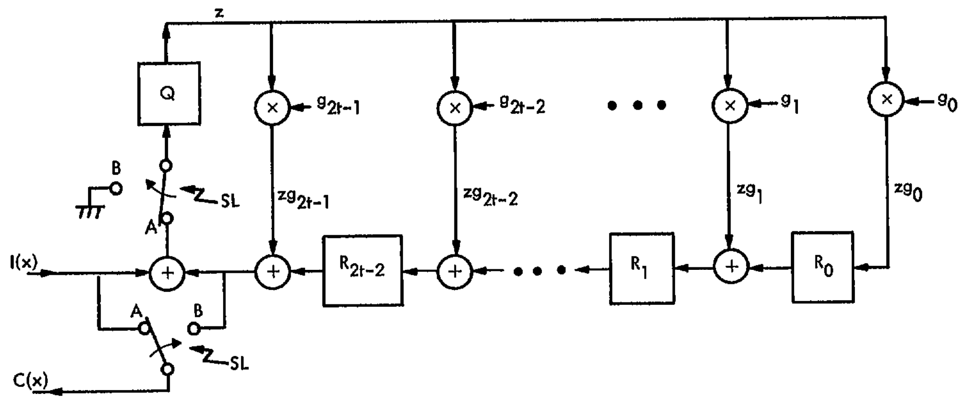

Fig. 1. A structure of a *t*-error correcting RS-encoder

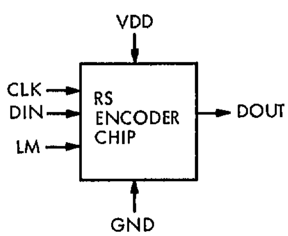

Fig. 2. Symbolic diagram of a RS encoder chip

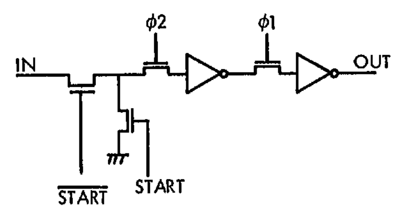

Fig. 3. Logic diagram of a 1-bit dynamic register with reset

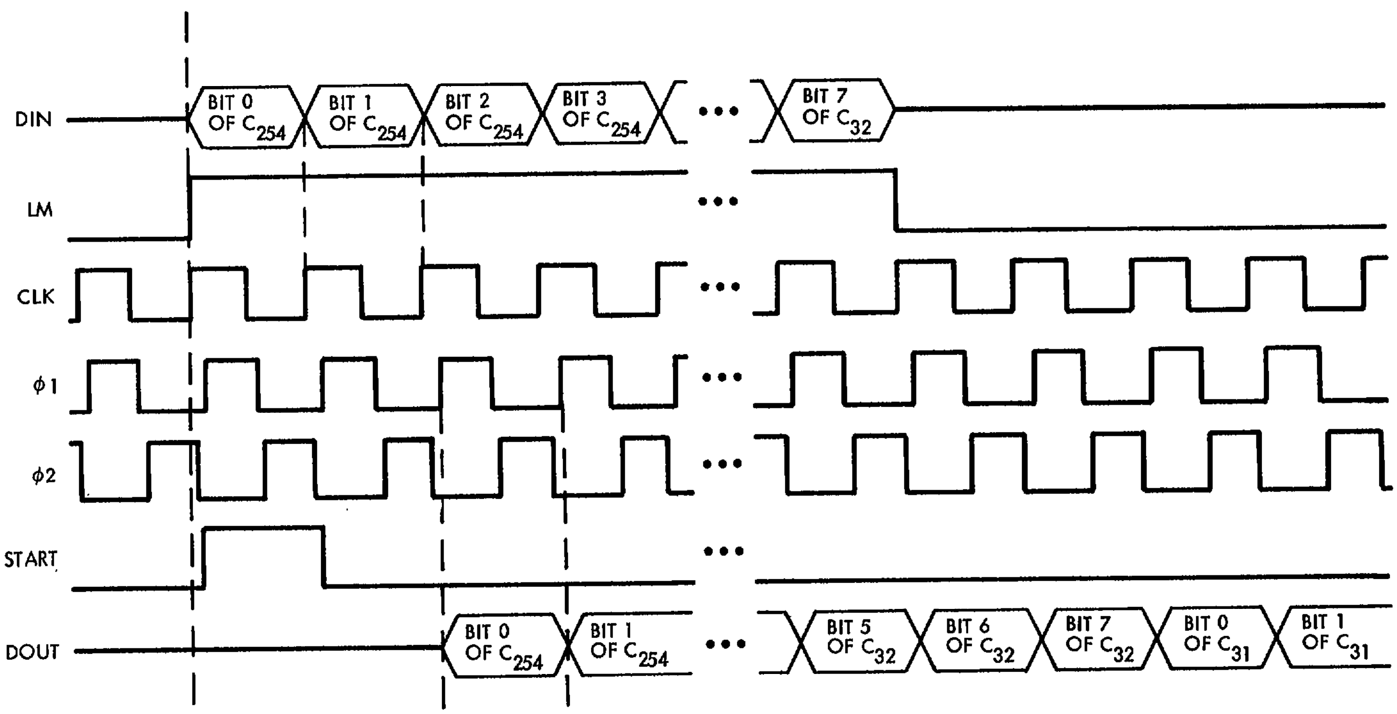

Fig. 4. The signals of DIN, LM, CLK, $\phi 1$, $\phi 2$, START, and DOUT

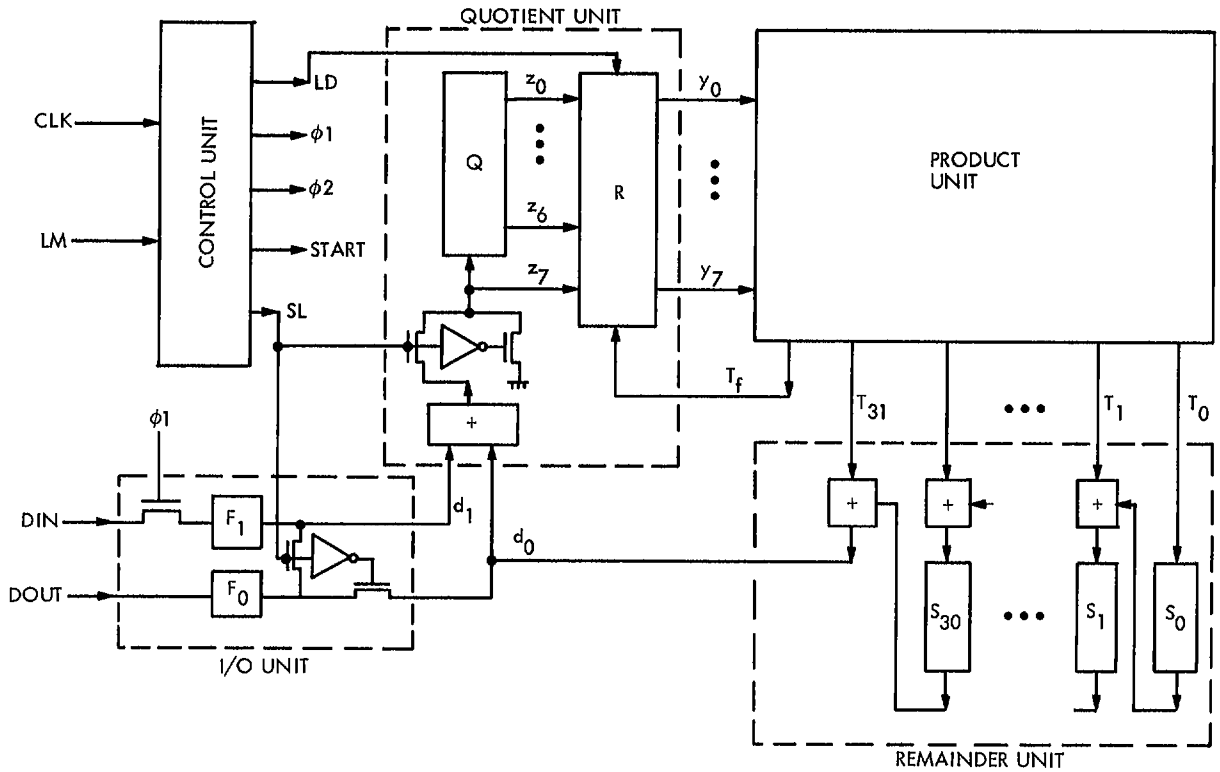

Fig. 5. Block diagram of a RS encoder

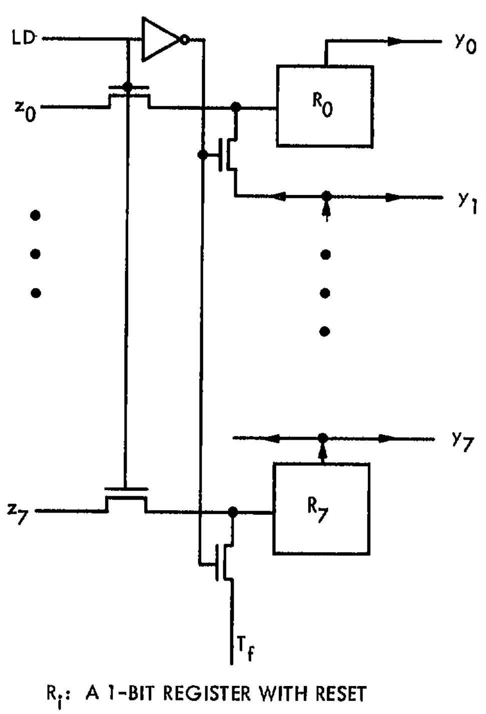

Fig. 6. Block diagram of register R

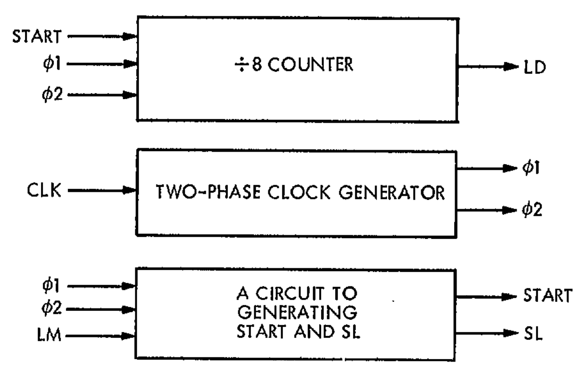

Fig. 7. Block diagram of the Control Unit

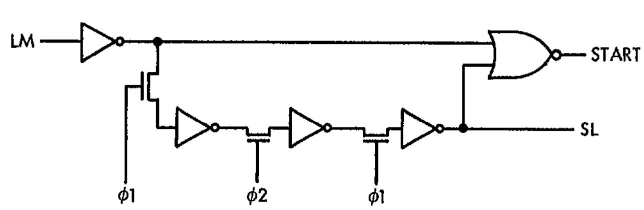

Fig. 8. Logic diagram of the circuit for generating control signals START and SL

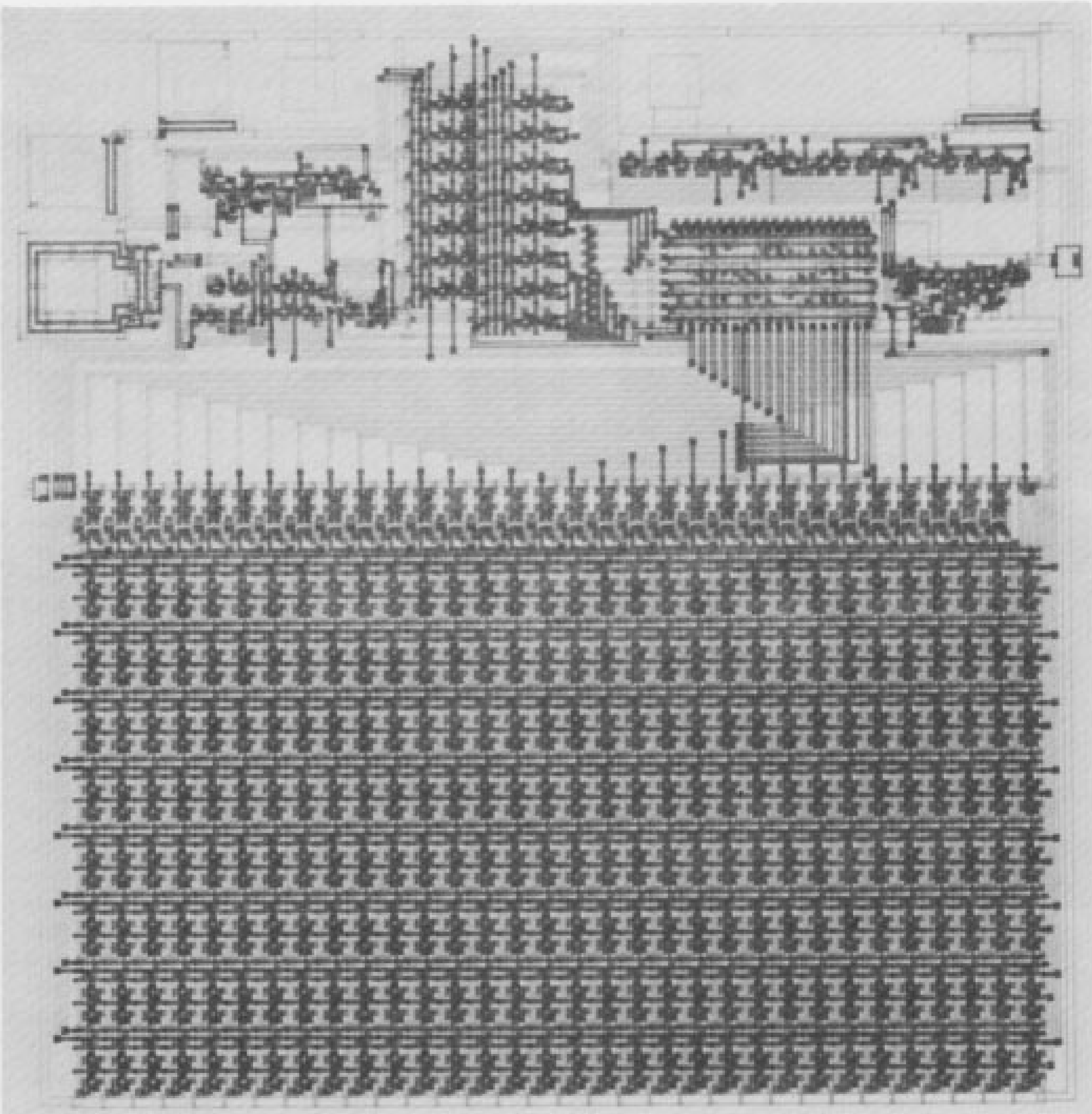

Fig. 9. Layout of the (255, 223) RS-encoder chip

Appendix A

In this appendix all 256 elements in $GF(2^8)$ are listed in Table A-1. These field elements are expressed in both the normal basis and its dual basis.

Table A-1. Representations of elements in $GF(2^8)$

Power j	Elements in normal base	$\mathrm{Tr}(\alpha^j)$	Elements in dual base	Power j	Elements in normal base	$\mathrm{Tr}(\alpha^j)$	Elements in dual base
*	00000000	0	00000000	35	11101000	0	00110111
0	00000001	0	01111111	36	01010111	0	01101110
1	00000010	1	11111111	37	10101110	1	11011100
2	00000100	1	11111110	38	11011011	1	10111000
3	00001000	1	11111101	39	00110001	0	01110000
4	00010000	1	11111010	40	01100010	1	11100000
5	00100000	1	11110101	41	11000100	1	11000001
6	01000000	1	11101010	42	00001111	1	10000011
7	10000000	1	11010101	43	00011110	0	00000110
8	10000111	1	10101011	44	00111100	0	00001100
9	10001001	0	01010111	45	01111000	0	00011000
10	10010101	1	10101110	46	11110000	0	00110000
11	10101101	0	01011100	47	01100111	0	01100001
12	11011101	1	10111001	48	11001110	1	11000011
13	00111101	0	01110011	49	00011011	1	10000111
14	01111010	1	11100111	50	00110110	0	00001110
15	11110100	1	11001110	51	01101100	0	00011100
16	01101111	1	10011100	52	11011000	0	00111000
17	11011110	0	00111001	53	00110111	0	01110001
18	00111011	0	01110010	54	01101110	1	11100011
19	01110110	1	11100100	55	11011100	1	11000110
20	11101100	1	11001001	56	00111111	1	10001100
21	01011111	1	10010011	57	01111110	0	00011001
22	10111110	0	00100110	58	11111100	0	00110011
23	11111011	0	01001101	59	01111111	0	01100110
24	01110001	1	10011010	60	11111110	1	11001100
25	11100010	0	00110101	61	01111011	1	10011000
26	01000011	0	01101010	62	11110110	0	00110001
27	10000110	1	11010100	63	01101011	0	01100010
28	10001011	1	10101000	64	11010110	1	11000100
29	10010001	0	01010000	65	00101011	1	10001000
30	10100101	1	10100001	66	01010110	0	00010001
31	11001101	0	01000011	67	10101100	0	00100011
32	00011101	1	10000110	68	11011111	0	01000110
33	00111010	0	00001101	69	00111001	1	10001101
34	01110100	0	00011011	70	01110010	0	00011010

Table A-1 (contd)

Power j	Elements in normal base	$\mathrm{Tr}(\alpha^j)$	Elements in dual base	Power j	Elements in normal base	$\mathrm{Tr}(\alpha^j)$	Elements in dual base
71	11100100	0	00110100	112	01000111	1	10010100
72	01001111	0	01101001	113	10001110	0	00101001
73	10011110	1	11010011	114	10011011	0	01010010
74	10111011	1	10100111	115	10110001	1	10100101
75	11110001	0	01001111	116	11100101	0	01001011
76	01100101	1	10011110	117	01001101	1	10010110
77	11001010	0	00111101	118	10011010	0	00101101
78	00010011	0	01111010	119	10110011	0	01011010
79	00100110	1	11110100	120	11100001	1	10110101
80	01001100	1	11101001	121	01000101	0	01101011
81	10011000	1	11010010	122	10001010	1	11010111
82	10110111	1	10100100	123	10010011	1	10101111
83	11101001	0	01001000	124	10100001	0	01011111
84	01010101	1	10010001	125	11000101	1	10111110
85	10101010	0	00100010	126	00001101	0	01111100
86	11010011	0	01000101	127	00011010	1	11111000
87	00100001	1	10001010	128	00110100	1	11110001
88	01000010	0	00010101	129	01101000	1	11100010
89	10000100	0	00101011	130	11010000	1	11000101
90	10001111	0	01010110	131	00100111	1	10001011
91	10011001	1	10101101	132	01001110	0	00010110
92	10110101	0	01011011	133	10011100	0	00101100
93	11101101	1	10110110	134	10111111	0	01011001
94	01011101	0	01101100	135	11111001	1	10110010
95	10111010	1	11011000	136	01110101	0	01100100
96	11110011	1	10110000	137	11101010	1	11001000
97	01100001	0	01100000	138	01010011	1	10010000
98	11000010	1	11000000	139	10100110	0	00100001
99	00000011	1	10000000 λ_0	140	11001011	0	01000010
100	00000110	0	00000001 λ_7	141	00010001	1	10000101
101	00001100	0	00000011	142	00100010	0	00001010
102	00011000	0	00000111	143	01000100	0	00010100
103	00110000	0	00001111	144	10001000	0	00101000
104	01100000	0	00011111	145	10010111	0	01010001
105	11000000	0	00111111	146	10101001	1	10100010
106	00000111	0	01111110	147	11010101	0	01000100
107	00001110	1	11111100	148	00101101	1	10001001
108	00011100	1	11111001	149	01011010	0	00010010
109	00111000	1	11110010	150	10110100	0	00100100
110	01110000	1	11100101	151	11101111	0	01001001
111	11100000	1	11001010	152	01011001	1	10010010

Table A-1 (contd)

Power j	Elements in normal base	$Tr(\alpha^j)$	Elements in dual base	Power j	Elements in normal base	$Tr(\alpha^j)$	Elements in dual base
153	10110010	0	00100101	194	01001001	0	01101000
154	11100011	0	01001010	195	10010010	1	11010000
155	01000001	1	10010101	196	10100011	1	10100000
156	10000010	0	00101010	197	11000001	0	01000000 λ_1
157	10000011	0	01010101	198	00000101	1	10000001
158	10000001	1	10101010	199	00001010	0	00000010 λ_6
159	10000101	0	01010100	200	00010100	0	00000100 λ_5
160	10001101	1	10101001	201	00101000	0	00001000 λ_4
161	10011101	0	01010011	202	01010000	0	00010000 λ_3
162	10111101	1	10100110	203	10100000	0	00100000 λ_2
163	11111101	0	01001100	204	11000111	0	01000001
164	01111101	1	10011001	205	00001001	1	10000010
165	11111010	0	00110010	206	00010010	0	00000101
166	01110011	0	01100101	207	00100100	0	00001011
167	11100110	1	11001011	208	01001000	0	00010111
168	01001011	1	10010111	209	10010000	0	00101111
169	10010110	0	00101110	210	10100111	0	01011110
170	10101011	0	01011101	211	11001001	1	10111101
171	11010001	1	10111010	212	00010101	0	01111011
172	00100101	0	01110100	213	00101010	1	11110111
173	01001010	1	11101000	214	01010100	1	11101110
174	10010100	1	11010001	215	10101000	1	11011101
175	10101111	1	10100011	216	11010111	1	10111011
176	11011001	0	01000111	217	00101001	0	01110111
177	00110101	1	10001110	218	01010010	1	11101111
178	01101010	0	00011101	219	10100100	1	11011110
179	11010100	0	00111011	220	11001111	1	10111100
180	00101111	0	01110110	221	00011001	0	01111000
181	01011110	1	11101100	222	00110010	1	11110000
182	10111100	1	11011001	223	01100100	1	11100001
183	11111111	1	10110011	224	11001000	1	11000010
184	01111001	0	01100111	225	00010111	1	10000100
185	11110010	1	11001111	226	00101110	0	00001001
186	01100011	1	10011111	227	01011100	0	00010011
187	11000110	0	00111110	228	10111000	0	00100111
188	00001011	0	01111101	229	11110111	0	01001110
189	00010110	1	11111011	230	01101001	1	10011101
190	00101100	1	11110110	231	11010010	0	00111010
191	01011000	1	11101101	232	00100011	0	01110101
192	10110000	1	11011010	233	01000110	1	11101011
193	11100111	1	10110100	234	10001100	1	11010110

Table A-1 (contd)

Power j	Elements in normal base	$Tr(\alpha^j)$	Elements in dual base	Power j	Elements in normal base	$Tr(\alpha^j)$	Elements in dual base
235	10011111	1	10101100	245	01111100	1	11100110
236	10111001	0	01011000	246	11111000	1	11001101
237	11110101	1	10110001	247	01110111	1	10011011
238	01101101	0	01100011	248	11101110	0	00110110
239	11011010	1	11000111	249	01011011	0	01101101
240	00110011	1	10001111	250	10110110	1	11011011
241	01100110	0	00011110	251	11101011	1	10110111
242	11001100	0	00111100	252	01010001	0	01101111
243	00011111	0	01111001	253	10100010	1	11011111
244	00111110	1	11110011	254	11000111	1	10111111

Appendix B

The binary mapping matrix for $\gamma = \alpha^{11}$ of the (255, 223) RS-encoder is given by

$$
\begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \\ T_7 \\ T_8 \\ T_9 \\ T_{10} \\ T_{11} \\ T_{12} \\ T_{13} \\ T_{14} \\ T_{15} \\ T_{16} \end{bmatrix}
=
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}
\begin{bmatrix} Z_0 \\ Z_1 \\ Z_2 \\ Z_3 \\ Z_4 \\ Z_5 \\ Z_6 \\ Z_7 \end{bmatrix}
$$